Mobile Intel® Pentium® 4 Processor with 533 MHz System Bus

Specification Update

September 2003

Notice: The mobile Intel® Pentium® 4 processor with 533 MHz system bus may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are documented in this Specification Update.

Document Number: 253176-004

Contents

Revision History

Version	Description	Date of Revision
-001	Initial Release	June 2003
-002	Added Erratum Z29.	July 2003
-003	Update Errata Z25. Added Errata Z30.	August 2003
-004	Added Errata Z31, Z32, Z33, and Z34.	September 2003

Preface

This document is an update to the specifications contained in the documents listed in the following Affected Documents/Related Documents table. It is a compilation of device and document errata, specification clarifications, and specification changes and is intended for hardware system manufacturers and for software developers of applications, operating system, and tools.

Information types defined in the Nomenclature section of this document are consolidated into this update document and are no longer published in other documents. This document may also contain information that has not been previously published.

Affected Documents

Document Title	Document Number
Mobile Intel® Pentium® 4 Processor with 533 MHz System Bus Datasheet	253028

Related Documents

Document Title	Document Number
Intel® Architecture Software Developer's Manual, Volume 1: Basic Architecture	http://developer.intel.com/design/pentium4/manuals/245470.htm
Intel® Architecture Software Developer's Manual, Volume 2: Instruction Set Reference	http://developer.intel.com/design/pentium4/manuals/245471.htm
Intel® Architecture Software Developer's Manual, Volume 3: System Programming Guide	http://developer.intel.com/design/pentium4/manuals/245472.htm

Nomenclature

S-Spec Number is a five-digit code used to identify products. Products are differentiated by their unique characteristics, e.g., core speed, L2 cache size, package type, etc. as described in the processor identification information table. Care should be taken to read all notes associated with each S-Spec number

Errata are design defects or errors. Errata may cause the mobile Intel® Pentium® processor's behavior to deviate from published specifications. Hardware and software designed to be used with a-y given stepping must assume that all errata documented for that stepping are present on all devices.

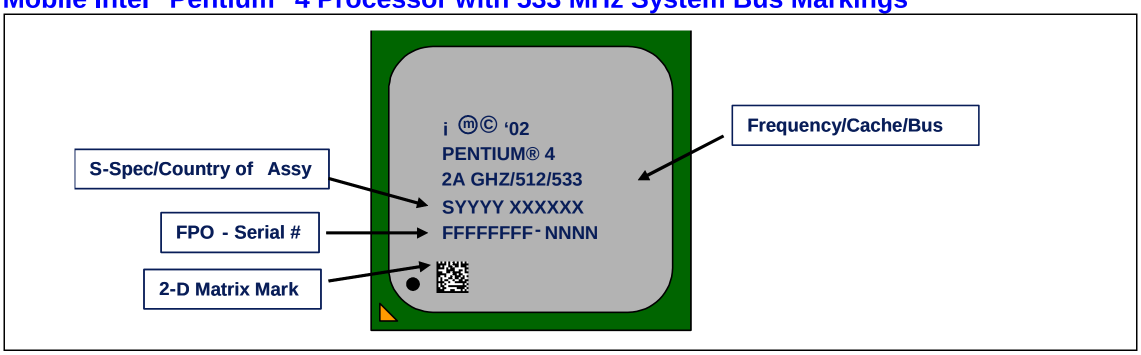
Specification Changes are modifications to the current published specifications. These changes will be incorporated in the next release of the specifications.

Specification Clarifications describe a specification in greater detail or further highlight a specification's impact to a complex design situation. These clarifications will be incorporated in the next release of the specifications.

Documentation Changes include typos, errors, or omissions from the current published specifications. These changes will be incorporated in the next release of the specifications.

General Information

Figure 1. Mobile Intel® Pentium® 4 Processor with 533 MHz System Bus Markings



Identification Information

The mobile Intel® Pentium® 4 processor can be identified by the following values:

Family[1]	Model[2]	Brand ID[3]
1111	0010	00001110
1111	0010	00001111

NOTES:

1. The Family corresponds to bits [11:8] of the EDX register after RESET, bits [11:8] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the generation field of the Device ID register accessible through Boundary Scan.
2. The Model corresponds to bits [7:4] of the EDX register after RESET, bits [7:4] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the model field of the Device ID register accessible through Boundary Scan.
3. The Brand ID corresponds to bits [7:0] of the EBX register after the CPUID instruction is executed with a 1 in the EAX register.

Table 2. Mobile Intel Pentium 4 Processor Identification Information

S-Spec	Core Stepping	L2 Cache Size (bytes)	Processor Signature	Core Frequency Perf Mode/ Batt Mode	Bus Freq	Voltage Perf Mode (Max)/Batt Mode	Package and Revision	Notes
SL723	D1	512K	0F29	2.40GHz/1.60GHz	533 MHz	1.525V/1.200V	31.0 mm FC rev 1.0	1
SL724	D1	512K	0F29	2.66GHz/1.60GHz	533 MHz	1.525V/1.200V	31.0 mm FC rev 1.0	1
SL725	D1	512K	0F29	2.80GHz/1.60GHz	533 MHz	1.525V/1.200V	31.0 mm FC rev 1.0	1
SL726	D1	512K	0F29	3.06GHz/1.60GHz	533 MHz	1.550V/1.200V	31.0 mm FC rev 1.0	1

NOTE: These parts are multiple VIDs

Summary Tables of Changes

The following table indicates the Errata, Documentation Changes, Specification Clarifications, or Specification Changes that apply to Intel Pentium 4 processors. Intel intends to fix some of the errata in a future stepping of the component, and to account for the other outstanding issues through documentation or specification changes as noted. This table uses the following notations:

Codes Used in Summary Table

Stepping

X:	Erratum, Specification Change or Clarification that applies to this stepping.
(No mark) or (Blank Box):	This erratum is fixed in listed stepping or specification change does not apply to listed stepping.

Status

Doc:	Document change or update that will be implemented.
PlanFix:	This erratum may be fixed in a future stepping of the product.
Fixed:	This erratum has been previously fixed.
NoFix:	There are no plans to fix this erratum.
PKG:	This column refers to errata on the the Intel® Pentium® 4 processor substrate.
AP:	APIC related erratum.
Shaded:	This item is either new or modified from the previous version of the document.

Note: Each Specification Update item is prefixed with a capital letter to distinguish the product. The key below details the letters that are used in Intel's microprocessor Specification Updates:

A = Intel® Pentium® II processor
B = Mobile Intel® Pentium® II processor
C = Intel® Celeron® processor
D = Intel® Pentium® II Xeon™ processor
E = Intel® Pentium® III processor
G = Intel® Pentium® III Xeon™ processor
H = Mobile Intel® Celeron® processor at 466 MHz, 433 MHz, 400 MHz, 366 MHz, 333 MHz, 300 MHz, and 266 MHz
K = Mobile Intel® Pentium® III Processor - M
M = Mobile Intel® Celeron® processor
N = Intel® Pentium® 4 processor
O = Intel® Xeon™ processor MP
P = Intel® Xeon™ processor
T = Mobile Intel® Pentium® 4 processor – M
V = Intel® Celeron® processor in the 478-Pin Package
Y = Intel® Pentium® M processor
Z = Mobile Intel® Pentium® 4 Processor with 533 MHz System Bus

NO.	D1	PLANS	ERRATA
Z1	X	No Fix	I/O restart in SMM may fail after simultaneous machine check exception (MCE)
Z2	X	No Fix	MCA registers may contain invalid information if RESET# occurs and PWRGOOD is not held asserted
Z3	X	No Fix	Transaction is not retried after BINIT#
Z4	X	No Fix	Invalid opcode 0FFFh requires a ModRM byte
Z5	X	No Fix	FSW may not be completely restored after page fault on FRSTOR or FLDENV instructions
Z6	X	No Fix	The processor flags #PF instead of #AC on an unlocked CMPXCHG8B instruction
Z7	X	No Fix	When in no-fill mode the memory type of large pages are incorrectly forced to uncacheable
Z8	X	No Fix	Processor may hang due to speculative page walks to non-existent system memory
Z9	X	No Fix	IA32_MC0_STATUS register overflow bit not set correctly
Z10		No Fix	MCA error code field in IA32_MC0_STATUS register may become out of sync with the rest of the register
Z11	X	No Fix	The IA32_MC1_STATUS register may contain incorrect information for correctable errors
Z12	X	No Fix	Debug mechanisms may not function as expected
Z13	X	No Fix	Machine check architecture error reporting and recovery may not work as expected
Z14	X	No Fix	Cascading of performance counters does not work correctly when forced overflow is enabled
Z15	X	No Fix	EMON event counting of x87 loads may not work as expected
Z16		Plan Fix	Buffer on resistance may exceed specification
Z17	X	No Fix	Processor issues inconsistent transaction size attributes for locked operation
Z18	X	No Fix	When the processor is in the System Management Mode (SMM), debug registers may be fully writeable
Z19	X	No Fix	Associated counting logic must be configured when using Event Selection Control (ESCR) MSR
Z20	X	No Fix	IA32_MC0_ADDR and IA32_MC0_MISC registers will contain invalid or stale data following a Data, Address, or Response Parity Error
Z21	X	No Fix	Processor may hang under certain frequencies and 12.5% STPCLK# duty cycle
Z22	X	No Fix	System may hang if a fatal cache error causes Bus Write Line (BWL) transaction to occur to the same cache line address as an outstanding Bus Read Line (BRL) or Bus Read-Invalidate Line (BRIL)
Z23	X	No Fix	Simultaneous assertion of A20M# and INIT# may result in incorrect data fetch
Z24	X	No Fix	A Write to APIC Registers Sometimes May Appear to Have Not Occurred
Z25	X	No Fix	Stop-Clock Assertion May Cause a System to Hang
Z26	X	No Fix	Parity Error in the L1 Cache may Cause the Processor to Hang
Z27	X	No Fix	Disabling a Local APIC Disables Both Logical Processor APICs on a Hyper-Threading Technology Enabled Processor
Z28	X	No Fix	STPCLK Throttling and Executing Code from Very Slow Memory Could Lead to a System Hang

NO.	D1	PLANS	ERRATA
Z29	X	No Fix	The State of the Resume Flag (RF Flag) in a Task-State Segment (TSS) May be Incorrect
Z30	X	No Fix	Changes to CR3 Register do not Fence Pending Instruction Page Walks
Z31	X	Plan Fix	Simultaneous Page Faults at Similar Page Offsets on Both Logical Processors of a Hyper-Threading Technology Enabled Processor May Cause Application Failure
Z32	X	No Fix	System Bus Interrupt Messages without Data that Receive a HardFailure Response May Hang the Processor
Z33	X	No Fix	Memory Type of the Load Lock Different from its Corresponding Store Unlock
Z34	X	No Fix	Shutdown and IERR# May Result Due to a Machine Check Exception on a Hyper-Threading Technology Enabled Processor

NOTE: For these steppings, this erratum may be worked around in BIOS.

NO	D1	STATUS	SPECIFICATION CHANGE
			There are no Specification Changes in this Specification Update revision.

NO.	D1	STATUS	SPECIFICATION CLARIFICATIONS
Z1	X	Doc	Clarifying the Behavior of the Time-Stamp Counter (TSC)

NO.	DOCUMENT REVISION	STATUS	DOCUMENTATION CHANGES
			There are no Documentation Changes in this Specification Update revision.

Errata

Z1. I/O Restart in SMM May Fail after Simultaneous Machine Check Exception (MCE)

Problem: If an I/O instruction (IN, INS, REP INS, OUT, OUTS, or REP OUTS) is being executed, and if the data for this instruction becomes corrupted, the processor will signal a Machine Check Exception (MCE). If the instruction is directed at a device that is powered down, the processor may also receive an assertion of SMI#. Since MCEs have higher priority, the processor will call the MCE handler, and the SMI# assertion will remain pending. However, upon attempting to execute the first instruction of the MCE handler, the SMI# will be recognized and the processor will attempt to execute the SMM handler. If the SMM handler is completed successfully, it will attempt to restart the I/O instruction, but will not have the correct machine state, due to the call to the MCE handler.

Implication: A simultaneous MCE and SMI# assertion may occur for one of the I/O instructions above. The SMM handler may attempt to restart such an I/O instruction, but will have an incorrect state due to the MCE handler call, leading to failure of the restart and shutdown of the processor.

Workaround: If a system implementation must support both SMM and board I/O restart, the first thing the SMM handler code should do is check for a pending MCE. If there is an MCE pending, the SMM handler should immediately exit via an RSM instruction and allow the MCE handler to execute. If there is no MCE pending, the SMM handler may proceed with its normal operation.

Status: For the steppings affected, see the *Summary Table of Changes*.

Z2. MCA Registers May Contain Invalid Information If RESET# Occurs and PWRGOOD Is Not Held Asserted

Problem: This erratum can occur as a result either of the following events:

Example 0PWRGOOD is de-asserted during a RESET# assertion causing internal glitches that may result in the possibility that the MCA registers latch invalid information.

Example 1Or during a reset sequence if the processor's power remains valid regardless of the state of PWRGOOD, and RESET# is re-asserted before the processor has cleared the MCA registers, the processor will begin the reset process again but may not clear these registers.

Implication: When this erratum occurs, the information in the MCA registers may not be reliable.

Workaround: Ensure that PWRGOOD remains asserted throughout any RESET# assertion and that RESET# is not re-asserted while PWRGOOD is de-asserted.

Status: For the steppings affected, see the *Summary Table of Changes*.

Z3. Transaction Is Not Retried after BINIT#

Problem: If the first transaction of a locked sequence receives a HITM# and DEFER# during the snoop phase it should be retried and the locked sequence restarted. However, if BINIT# is also asserted during this transaction, it will not be retried.

Implication: When this erratum occurs, locked transactions will unexpectedly not be retried.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

Z4. Invalid Opcode 0FFFh Requires a ModRM Byte

Problem: Some invalid opcodes require a ModRM byte (or other following bytes), while others do not. The invalid opcode 0FFFh did not require a ModRM byte in previous generation Intel architecture processors, but does in the Pentium 4 processor.

Implication: The use of an invalid opcode 0FFFh without the ModRM byte may result in a page or limit fault on the Pentium 4 processor.

Workaround: Use a ModRM byte with invalid 0FFFh opcode.

Status: For the steppings affected, see the *Summary Table of Changes*.

Z5. FSW May Not Be Completely Restored after Page Fault on FRSTOR or FLDENV Instructions

Problem: If the FPU operating environment or FPU state (operating environment and register stack) being loaded by an FLDENV or FRSTOR instruction wraps around a 64-KB or 4-GB boundary and a page fault (#PF) or segment limit fault (#GP or #SS) occurs on the instruction near the wrap boundary, the upper byte of the FPU status word (FSW) might not be restored. If the fault handler does not restart program execution at the faulting instruction, stale data may exist in the FSW.

Implication: When this erratum occurs, stale data will exist in the FSW.

Workaround: Ensure that the FPU operating environment and FPU state do not cross 64-KB or 4-GB boundaries. Alternately, ensure that the page fault handler restarts program execution at the faulting instruction after correcting the paging problem.

Status: For the steppings affected, see the *Summary Table of Changes*.

Z6. The Processor Flags #PF Instead of #AC on an Unlocked CMPXCHG8B Instruction

Problem: If a data page fault (#PF) and alignment check fault (#AC) both occur for an unlocked CMPXCHG8B instruction, then #PF will be flagged.

Implication: Software that depends #AC before #PF will be affected since #PF is flagged in this case.

Workaround: Remove the software’s dependency on the fact that #AC has precedence over #PF. Alternately, reload the page in the page fault handler and then restart the faulting instruction

Status: For the steppings affected, see the *Summary Table of Changes*.

Z7. When in No-Fill Mode the Memory Type of Large Pages Are Incorrectly Forced to Uncacheable

Problem: When the processor is operating in No-Fill Mode (CR0.CD=1), the paging hardware incorrectly forces the memory type of large (PSE-4M and PAE-2M) pages to uncacheable (UC) memory type regardless of the MTRR settings. By forcing the memory type of these pages to UC, load operations, which should hit valid data in the L1 cache, are forced to load the data from system memory. Some applications will lose the performance advantage associated with the caching permitted by other memory types.

Implication: This erratum may result in some performance degradation when using no-fill mode with large pages.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

Z8. Processor May Hang Due to Speculative Page Walks to Non-Existent System Memory

Problem: A load operation that misses the Data Translation Lookaside Buffer (DTLB) will result in a page-walk. If the page-walk loads the Page Directory Entry (PDE) from cacheable memory and that PDE load returns data that points to a valid Page Table Entry (PTE) in uncacheable memory the processor will access the address referenced by the PTE. If the address referenced does not exist the processor will hang with no response from system memory.

Implication: Processor may hang due to speculative page walks to non-existent system memory.

Workaround: Page directories and page tables in UC memory space which are marked valid must point to physical addresses that will return a data response to the processor.

Status: For the steppings affected, see the *Summary Table of Changes*.

Z9. IA32_MC0_STATUS Register Overflow Bit Not Set Correctly

Problem: The Overflow Error bit (bit 62) in the IA32_MC0_STATUS register indicates, when set, that a machine check error occurred while the results of a previous error were still in the error reporting bank (i.e. the valid bit was set when the new error occurred). In the case of this erratum, if an uncorrectable error is logged in the error-reporting bank and another error occurs, the overflow bit will not be set.

Implication: When this erratum occurs the overflow bit will not be set.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

Z10. MCA Error Code Field in IA32_MC0_STATUS Register May become out of Sync with the Rest of the Register

Problem: The MCA Error Code field of the IA32_MC0_STATUS register gets written by a different mechanism than the rest of the register. For uncorrectable errors, the other fields in the IA32_MC0_STATUS register are only updated by the first error. Any subsequent errors cause the Overflow Error bit to be asserted until this register is cleared. Because of this erratum, any further errors that are detected will update the MCA Error Code field without updating the rest of the register, thereby leaving the IA32_MC0_STATUS register with stale information.

Implication: When this erratum occurs, the IA32_MC0_STATUS register contains stale information.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

Z11. The IA32_MC1_STATUS Register May Contain Incorrect Information for Correctable Errors

Problem: When a speculative load operation hits the L2 cache and receives a correctable error, the IA32_MC1_STATUS register may be updated with incorrect information. The IA32_MC1_STATUS register should not be updated for speculative loads.

Implication: When this erratum occurs, the IA32_MC1_STATUS register will contain incorrect information for correctable errors.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

Z12. Debug Mechanisms May Not Function As Expected

Problem: Certain debug mechanisms may not function as expected on the processor. The cases are as follows:

- When the following conditions occur: 1) An FLD instruction signals a stack overflow or underflow, 2) the FLD instruction splits a page-boundary or a 64 byte cache line boundary, 3) the instruction matches a Debug Register on the high page or cache line respectively, and 4) the FLD has a stack fault and a memory fault on a split access, the processor will only signal the stack fault and the debug exception will not be taken.
- When a data breakpoint is set on the ninth and/or tenth byte(s) of a floating point store using the Extended Real data type, and an unmasked floating point exception occurs on the store, the break point will not be captured.
- When any instruction has multiple debug register matches, and any one of those debug registers is enabled in DR7, all of the matches should be reported in DR6 when the processor goes to the debug handler. This is not true during a REP instruction. As an example, during execution of a REP MOVSW instruction the first iteration a load matches DR0 and DR2 and sets DR6 as FFFF0FF5h. On a subsequent iteration of the instruction, a load matches only DR0. The DR6 register is expected to still contain FFFF0FF5h, but the processor will update DR6 to FFFF0FF1h.
- A data breakpoint that is set on a load to uncacheable memory may be ignored due to an internal segment register access conflict. In this case the system will continue to execute instructions, bypassing the intended breakpoint. Avoiding having instructions that access segment descriptor registers, e.g., LGDT, LIDT close to the UC load, and avoiding serialized instructions before the UC load will reduce the occurrence of this erratum.

Implication: Certain debug mechanisms do not function as expected on the processor.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

Z13. Machine Check Architecture Error Reporting and Recovery May Not Work As Expected

Problem: When the processor detects errors it should attempt to report and/or recover from the error. In the situations described below, the processor does not report and/or recover from the error(s) as intended.

- When a transaction is deferred during the snoop phase and subsequently receives a Hard Failure response, the transaction should be removed from the bus queue so that the processor may proceed. Instead, the transaction is not properly removed from the bus queue, the bus queue is blocked, and the processor will hang.
- When a hardware prefetch results in an uncorrectable tag error in the L2 cache, IA32_MC0_STATUS.UNCOR and IA32_MC0_STATUS.PCC are set but no Machine Check Exception (MCE) is signaled. No data loss or corruption occurs because the data being prefetched has not been used. If the data location with the uncorrectable tag error is subsequently accessed, an MCE will occur. However, upon this MCE, or any other subsequent MCE, the information for that error will not be logged because IA32_MC0_STATUS.UNCOR has already been set and the MCA status registers will not contain information about the error which caused the MCE assertion but instead will contain information about the prefetch error event.

- When the reporting of errors is disabled for Machine Check Architecture (MCA) Bank 2 by setting all IA32_MC2_CTL register bits to 0, uncorrectable errors should be logged in the IA32_MC2_STATUS register but no machine-check exception should be generated. Uncorrectable loads on bank 2, which would normally be logged in the IA32_MC2_STATUS register, are not logged.
- When one half of a 64 byte instruction fetch from the L2 cache has an uncorrectable error and the other 32 byte half of the same fetch from the L2 cache has a correctable error, the processor will attempt to correct the correctable error but cannot proceed due to the uncorrectable error. When this occurs the processor will hang.
- When an L1 cache parity error occurs, the cache controller logic should write the physical address of the data memory location that produced that error into the IA32_MC1_ADDR register. In some instances of a parity error on a load operation that hits the L1 cache, however, the cache controller logic may write the physical address from a subsequent load or store operation into the IA32_MC1_ADDR register.
- The local xAPIC has an Error Status Register which records all errors which it detects. Bit 6 of this register, the Receive Illegal Vector bit, is set when the local xAPIC detects an illegal vector in a message that it received. When an illegal vector error is received on the same internal clock that the error status register is being written due to a previous error, bit 6 does not get set and illegal vector errors are not flagged.
- If an instruction fetch results in an uncorrectable error and there is also a debug breakpoint at this address, the processor will livelock and the uncorrectable error will not be logged in the machine check registers.
- The MCA Overflow bit should be set when an uncorrectable error resides within the register bank (valid bit is already set) and any subsequent errors occur. The Overflow bit being set indicates that more than one error has occurred. Because of this erratum, if any further errors occur, the MCA Overflow bit will not be updated; thereby incorrectly indicating only one error has been received.

Implication: The processor is unable to correctly report and/or recover from certain errors.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

Z14. Cascading of Performance Counters Does Not Work Correctly When Forced Overflow Is Enabled

Problem: The performance counters are organized into pairs. When the CASCADE bit of the Counter Configuration Control Register (CCCR) is set, a counter that overflows will continue to count in the other counter of the pair. The FORCE_OVF bit forces the counters to overflow on every non-zero increment. When the FORCE_OVF bit is set, the counter overflow bit will be set but the counter no longer cascades.

Implication: The performance counters do not cascade when the FORCE_OVF bit is set.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

Z15. EMON Event Counting of x87 Loads May Not Work As Expected

Problem: If a performance counter is set to count x87 loads and floating point exceptions are unmasked, the FPU Operand Data Pointer (FDP) may become corrupted.

Implication: When this erratum occurs, the FPU Operand Data Pointer (FDP) may become corrupted.

Workaround: This erratum will not occur with floating point exceptions masked. If floating point exceptions are unmasked, then performance counting of x87 loads should be disabled.

Status: For the steppings affected, see the *Summary Table of Changes*.

Z16. Buffer on Resistance May Exceed Specification

Problem: The datasheet specifies the resistance range for R_{ON} (Buffer On Resistance) for the AGTL+ and Asynchronous GTL+ buffers as 5 to 11 Ohms. Due to this erratum, R_{ON} may be as high as 13.11 Ohms.

Implication: The R_{ON} value affects the voltage level of the signals when the buffer is driving the signal low. A higher R_{ON} may adversely affect the system's ability to meet specifications such as V_{IL}. As the system design also affects margin to specification, designs may or may not have sufficient margin to function properly with an increased R_{ON}. System designers should evaluate whether a particular system is affected by this erratum. Designs that follow the recommendations in the *Intel® Pentium® 4 Processor and Intel® 850 Chipset Platform Design Guide* are not expected to be affected.

Workaround: No workaround is necessary for systems with margin sufficient to accept a higher R_{ON}.

Status: For the steppings affected, see the *Summary Table of Change.*

Z17. Processor Issues Inconsistent Transaction Size Attributes for Locked Operation

Problem: When the processor is in the Page Address Extension (PAE) mode and detects the need to set the Access and/or Dirty bits in the page directory or page table entries, the processor sends an 8 byte load lock onto the system bus. A subsequent 8 byte store unlock is expected, but instead a 4 byte store unlock occurs. Correct data is provided since only the lower bytes change, however external logic monitoring the data transfer may be expecting an 8-byte store unlock.

Implication: No known commercially available chipsets are affected by this erratum.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

Z18. When the Processor Is in the System Management Mode (SMM), Debug Registers May Be Fully Writeable

Problem: When in System Management Mode (SMM), the processor executes code and stores data in the SMRAM space. When the processor is in this mode and writes are made to DR6 and DR7, the processor should block writes to the reserved bit locations. Due to this erratum, the processor may not block these writes. This may result in invalid data in the reserved bit locations.

Implication: Reserved bit locations within DR6 and DR7 may become invalid.

Workaround: Software may perform a read/modify/write when writing to DR6 and DR7 to ensure that the values in the reserved bits are maintained.

For the steppings affected, see the *Summary Table of Changes*.

Z19. Associated Counting Logic Must Be Configured When Using Event Selection Control (ESCR) MSR

Problem: ESCR MSRs allow software to select specific events to be counted, with each ESCR usually associated with a pair of performance counters. ESCRs may also be used to qualify the detection of at-retirement

events that support precise-event-based sampling (PEBS). A number of performance metrics that support PEBS require a 2nd ESCR to tag uops for the qualification of at-retirement events. (The first ESCR is required to program the at-retirement event.) Counting is enabled via counter configuration control registers (CCCR) while the event count is read from one of the associated counters. When counting logic is configured for the subset of at-retirement events that require a second ESCR to tag uops, at least one of the CCCRs in the same group of the second ESCR must be enabled.

Implication: If no CCCR/counter is enabled in a given group, the ESCR in that group that is programmed for tagging uops will have no effect. Hence a subset of performance metrics that require a second ESCR for tagging uops may result in 0 count.

Workaround: Ensure that at least one CCCR/counter in the same group as the tagging ESCR is enabled for those performance metrics that require two ESCRs and tagging uops for at-retirement counting.

Status: For the steppings affected, see the *Summary Table of Changes*.

Z20. IA32_MC0_ADDR and IA32_MC0_MISC Registers Will Contain Invalid or Stale Data Following a Data, Address, or Response Parity Error

Problem: If the processor experiences a data, address, or response parity error, the ADDRV and MISCV bits of the IA32_MC0_STATUS register are set, but the IA32_MC0_ADDR and IA32_MC0_MISC registers are not loaded with data regarding the error.

Implication: When this erratum occurs, the IA32_MC0_ADDR and IA32_MC0_MISC registers will contain invalid or stale data.

Workaround: Ignore any information in the IA32_MC0_ADDR and IA32_MC0_MISC registers after a data, address or response parity error.

Status: For the steppings affected, see the *Summary Table of Changes*.

Z21. Processor May Hang under Certain Frequencies and 12.5% STPCLK# Duty Cycle

Problem: If a system de-asserts STPCLK# at a 12.5% duty cycle, the processor is running below 2 GHz, and the processor thermal control circuit (TCC) on-demand clock modulation is active, the processor may hang. This erratum does not occur under the automatic mode of the TCC.

Implication: When this erratum occurs, the processor will hang.

Workaround: If use of the on-demand mode of the processor's TCC is desired in conjunction with STPCLK# modulation, then assure that STPCLK# is not asserted at a 12.5% duty cycle.

Status: For the steppings affected, see the *Summary Table of Changes*.

Z22. System May Hang if a Fatal Cache Error Causes Bus Write Line (BWL) Transaction to Occur to the Same Cache Line Address as an Outstanding Bus Read Line (BRL) or Bus Read-Invalidate Line (BRIL)

Problem: A processor internal cache fatal data ECC error may cause the processor to issue a BWL transaction to the same cache line address as an outstanding BRL or BRIL. As it is not typical behavior for a single processor to have a BWL and a BRL/BRIL concurrently outstanding to the same address, this may represent an unexpected scenario to system logic within the chipset.

Implication: The processor may not be able to fully execute the machine check handler in response to the fatal cache error if system logic does not ensure forward progress on the system bus under this scenario.

Workaround: System logic should ensure completion of the outstanding transactions. Note that during recovery from a fatal data ECC error, memory image coherency of the BWL with respect to BRL/BRIL transactions is not important. Forward progress is the primary requirement.

Status: For the steppings affected, see the *Summary Table of Changes*.

Z23. Simultaneous Assertion of A20M# and INIT# May Result in Incorrect Data Fetch

Problem: If A20M# and INIT# are simultaneously asserted by software, followed by a data access to the 0xFFFFFXXX memory region, with A20M# still asserted, incorrect data will be accessed. With A20M# asserted, an access to 0xFFFFFXXX should result in a load from physical address 0xFFEFFXXX. However, in the case of A20M# and INIT# being asserted together, the data load will actually be from the physical address 0xFFFFFXXX. Code accesses are not affected by this erratum.

Implication: Processor may fetch incorrect data, resulting in BIOS failure.

Workaround: Deasserting and reasserting A20M# prior to the data access will workaround this erratum.

Status: For the steppings affected, see the *Summary Table of Changes*.

Z24. A Write to APIC Registers Sometimes May Appear to Have Not Occurred

Problem: In respect to the retirement of instructions, stores to the uncacheable memory-based APIC register space are handled in a non-synchronized way. For example if an instruction that masks the interrupt flag, e.g. CLI, is executed soon after an uncacheable write to the Task Priority Register (TPR) that lowers the APIC priority, the interrupt masking operation may take effect before the actual priority has been lowered. This may cause interrupts whose priority is lower than the initial TPR, but higher than the final TPR, to not be serviced until the interrupt flag is finally cleared, i.e. by STI instruction. Interrupts will remain pending and are not lost.

Implication: In this example the processor may allow interrupts to be accepted but may delay their service.

Workaround: This non-synchronization can be avoided by issuing an APIC register read after the APIC register write. This will force the store to the APIC register before any subsequent instructions are executed. No commercial operating system is known to be impacted by this erratum.

Status: For the steppings affected, see the *Summary Table of Changes*.

Z25. Stop-Clock Assertion May Cause a System to Hang

Problem: Inappropriate assertion of the STPCLK# signal may cause a system to hang. The system may hang upon the following inappropriate assertions of the STPCLK# signal:

1. STPCLK# is asserted prior to the first time that a logical processor is awakened from the wait-for-SIPI State.

2. STPCLK_ACK bus cycle is deferred by an agent external to the processor for a period of time long enough for the chipset to deassert and then reassert STPCLK#. A processor supporting Hyper-Threading Technology may fail to detect the deassertion/reassertion and hence will not generate a STPCLK_ACK in response to the second STPCLK# assertion.

Implication: When this erratum occurs, the system may hang.

Workaround: BIOS should initialize the second thread of the processor supporting Hyper-Threading Technology prior to STPCLK# assertion. Additionally, it is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the Summary Table of Changes.

Z26. Parity Error in the L1 Cache May Cause the Processor to Hang

Problem: If a locked operation accesses a line in the L1 cache that has a parity error, it is possible that the processor may hang while trying to evict the line.

Implication: If this erratum occurs, it may result in a system hang. Intel has not observed this erratum with any commercially available software.

Workaround: None.

Status: For the steppings affected, see the *Summary Table of Changes*.

Z27. Disabling a Local APIC Disables Both Logical Processor APICs on a Hyper-Threading Technology Enabled Processor

Problem: Disabling a local APIC on one logical processor of a Hyper-Threading Technology enabled processor by clearing bit 11 of the IA32_APIC_BASE MSR will effectively disable the local APIC on the other logical processor.

Implication: Disabling a local APIC on one logical processor prevents the other logical processor from sending or receiving interrupts. Multiprocessor Specification compliant BIOSs and multiprocessor operating systems typically leave all local APICs enabled preventing any end-user visible impact from this erratum.

Workaround: Do not disable the local APICs in a Hyper-Threading Technology enabled processor.

Status: For the steppings affected, see *the Summary Table of Changes.*

Z28. STPCLK Throttling and Executing Code From Very Slow Memory Could Lead to a System Hang

Problem: The system may hang when the following conditions are met:

1) Periodic STPCLK mechanism is enabled via the chipset

2) Hyper-Threading Technology is enabled

3) One logical processor is waiting for an event (i.e. hardware interrupt)

4) The other logical processor executes code from very slow memory such that every code fetch is deferred long enough for the STPCLK to be re-asserted

Implication: If this erratum occurs, the processor will go into and out of the sleep state without making forward progress, as the logical processor will not be able to service any pending event. This erratum has not been observed in any commercial platform running commercial software.

Workaround: None

Status: For the steppings affected, see *the Summary Table of Changes.*

Z29. The State of the Resume Flag (RF Flag) in a Task-State Segment (TSS) May be Incorrect

Problem: After executing a JMP instruction to the next (or other) task through a hardware task switch, it is possible for the state of the RF flag (in the EFLAGS register image) to be incorrect.

Implication: The RF flag is normally used for code breakpoint management during debug of an application. It is not typically used during normal program execution. Code breakpoints or single step debug behavior in the presence of hardware task switches, therefore, may be unpredictable as a result of this erratum. This erratum has not been observed in commercially available software.

Workaround: None.

Status: For the steppings affected, see the Summary Table of Changes.

Z30. Changes to CR3 Register do not Fence Pending Instruction Page Walks

Problem: When software writes to the CR3 register, it is expected that all previous/outstanding code, data accesses and page walks are completed using the previous value in CR3 register. Due to this erratum, it is possible that a pending instruction page walk is still in progress, resulting in an access (to the PDE portion of the page table) that may be directed to an incorrect memory address.

Implication: The results of the access to the PDE will not be consumed by the processor so the return of incorrect data is benign. However, the system may hang if the access to the PDE does not complete with data (e.g. infinite number of retries).

Workaround: It is possible for the BIOS to have a workaround for this erratum.

Status: For the steppings affected, see the *Summary Table of Changes*.

Z31. Simultaneous Page Faults at Similar Page Offsets on Both Logical Processors of a Hyper-Threading Technology Enabled Processor May Cause Application Failure

Problem: An incorrect value of CR2 may be presented to one of the logical processors of an HT Technology enabled processor if a page access fault is encountered on one logical processor in the same clock cycle that the other logical processor also encounters a page fault. Both accesses must cross the same 4 byte aligned offset for this erratum to occur. Only a small percentage of such simultaneous accesses are vulnerable. The vulnerability of the alignment for any given fault is dependent on the state of other circuitry in the processor. Additionally, a third fault from an access that occurs sequentially after one of these simultaneous faults has to be pending at the time of the simultaneous faults. This erratum is caused by a one-cycle hole in the logic that controls the timing by which a logical processor is allowed to access an internal asynchronous fault address register. The end result is that the value of CR2 presented to one logical processor may be corrupted.

Implication: The operating system is likely to terminate the application that generated an incorrect value of CR2.

Workaround: An operating system or page management software can significantly reduce the already small possibility of encountering this failure by restarting or retrying the faulting instruction and only terminate the application on a subsequent failures of the same instruction. It is possible for BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Tables of Changes.*

Z32. System Bus Interrupt Messages without Data that Receive a HardFailure Response May Hang the Processor

Problem: When a system bus agent (processor or chipset) issues an interrupt transaction without data onto the system bus and the transaction receives a HardFailure response, a potential processor hang can occur.

The processor, which generates an inter-processor interrupt (IPI) that receives the HardFailure response, will still log the MCA error event cause as HardFailure, even if the APIC causes a hang. Other processors, which are true targets of the IPI, will also hang on hardfail-without-data, but will not record an MCA HardFailure event as the cause. If a HardFailure response occurs on a system bus interrupt message with data, the APIC will complete the operation so as not to hang the processor.

Implication: The processor may hang.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes.*

Z33. Memory Type of the Load Lock Different from its Corresponding Store Unlock

Problem: A use-once protocol is employed to ensure that the processor in a multi-agent system may access data that is loaded into its cache on a Read-for-Ownership operation at least once before it is snooped out by another agent. This protocol is necessary to avoid a multi-agent livelock scenario in which the processor cannot gain ownership of a line and modify it before that data is snooped out by another agent. In the case of this erratum, split load lock instructions incorrectly trigger the use-once protocol. A load lock operation accesses data that splits across a page boundary with both pages of WB memory type. The use-once protocol activates and the memory type for the split halves get forced to UC. Since use-once does not apply to stores, the store unlock instructions go out as WB memory type. The full sequence on the bus is: locked partial read (UC), partial read (UC), partial write (WB), locked partial write (WB). The use-once protocol should not be applied to load locks.

Implication: When this erratum occurs, the memory type of the load lock will be different than the memory type of the store unlock operation. This behavior (load locks and store unlocks having different memory types) does not introduce any functional failures such as system hangs or memory corruption.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes.*

Z34. Shutdown and IERR# May Result Due to a Machine Check Exception on a Hyper-Threading Technology Enabled Processor

Problem: When a Machine Check Exception (MCE) occurs due to an internal error, both logical processors on a Hyper-Threading Technology enabled processor normally vector to the MCE handler. However, if one of the logical processors is in the “Wait-for-SIPI” state, that logical processor will not have an MCE handler and will shut down and assert IERR#.

Implication: A processor with a logical processor in the “Wait-for-SIPI” state will shut down when an MCE occurs on the other thread.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Specification Changes

There are no Specification Changes in this Specification Update revision.

Specification Clarifications

The Specification Clarifications listed in this section apply to the following documents:

- *Mobile Intel® Pentium® 4 Processor with 533 MHz System Bus Datasheet*, Reference Number 253028.
- *IA-32 Intel(R) Architecture Software Developer's Manual* volumes 1, 2, and 3.

All Specification Clarifications will be incorporated into a future version of the appropriate Mobile Intel Pentium 4 processor documentation.

Z1. Clarifying the Behavior of the Time-Stamp Counter (TSC)

The behavior of the Time Stamp Counter measured across a period of time may differ due to power management features. This Specification Clarification applies to the following document: *IA-32 Intel® Architecture Software Developer's Manual,* Volume 3: System Programming Guide; Chapter 15: Debugging and Performance Monitoring; Section 15.7: Time-Stamp Counter.

New definition of the Time Stamp Counter will state:

The time-stamp counter (as implemented in the Pentium 4, Intel Xeon, P6 family, and Pentium processors) is a 64-bit counter that is set to 0 following the hardware reset of the processor. Following reset, the counter is incremented every processor clock cycle, even when the porcessor is halted by the HLT instruction or the external STPCLK# pin. However, the assertion of the external DPSLP# pin may cause the time-stamp counter to stop and Intel SpeedStep® technology transitions may cause the frequency at which the time-stamp counter increments to change in accordance with the processor's internal clock frequency.

Previous definition:

The time-stamp counter (as implemented in the Pentium 4, Intel Xeon, P6 family, and Pentium processors) is a 64-bit counter that is set to 0 following the hardware reset of the processor. Following reset, the counter is incremented every processor clock cycle, even when the processor is halted by the HLT instruction or the external STPCLK# pin.

Documentation Changes

There are no Documentation Changes in this Specification Update revision.